API Documentation
PGA 9500
OPC UA

Rego-Fix AG, Version: 1.1.6

This API (Application Programming Interface) documentation offers comprehensive guidance for the integration of the PGA 9500 powRgrip ® clamping unit into automation cells. It allows you to control various actions and retrieve important statistics, errors, maintenance information and notifications from the PGA 9500. Whether you are developing software, building automation systems, or troubleshooting the PGA unit, this documentation will assist you in understanding the API commands and their corresponding functionalities.

Get ready to explore the capabilities of the PGA communication interface and unlock the full efficiency potential of the PGA 9500.

Index

1. Introduction

The PGA 9500 enables the automated clamping of powRgrip ® toolholders and can be easily integrated into automation cells via OPC UA communication protocol. A useful HMI interface is available for commissioning and maintenance.

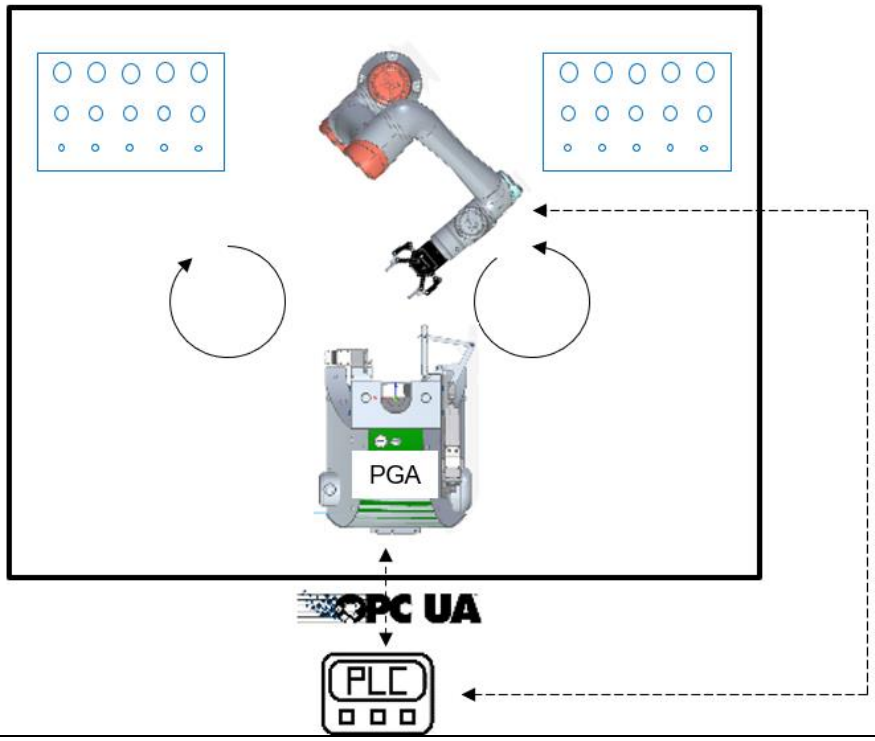

Fig.0: PGA 9500 in an automation cell

2. Tool-change handling

To supply the PGA 9500 with a cutting tool/collet/toolholder, an additional handling system is required which can be chosen by the system integrator. This could be a robot or an alternative feeding system. The controller of the handling system can communicate with the PGA 9500 via a network connection in the OPC UA architecture and control the commands of the PGA interface.

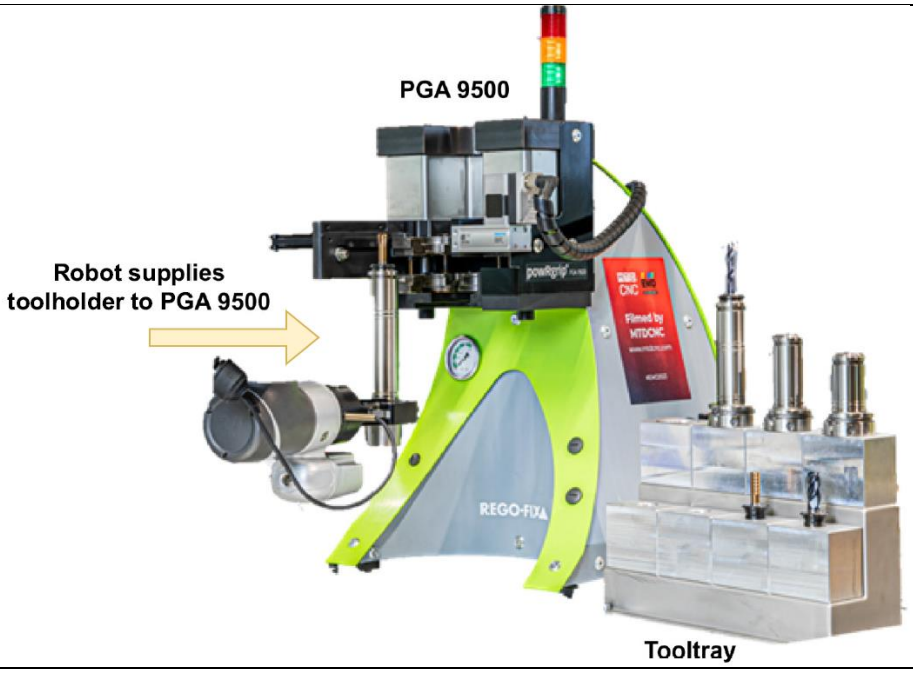

Fig.1: Visualization of the PGA tool-change handling

3. Operating modes

3.1. Automatic (default)

This mode allows all the necessary commands to be executed via the PGA interface. The automatic mode should be set by default, otherwise it would have to be set via the HMI interface to use the PGA in an automated system with all its functions and be able to execute them via the main controller with OPC UA protocol. In the event of a fault, the unit will stop and report the error via status code. The PGA is automatically initialized after enabling (setEnable) and is ready for use.

3.2. Manual

This mode can be set via the HMI interface. Once set, this mode must be confirmed with 'reset'. In this mode, all functions can be performed manually and separately via the HMI interface. For example, to check individual processes in the event of an error. Individual settings can also be made this way. You must start the process manually if the PGA is not initialized.

4. Connection setup

To establish a seamless communication link between the PGA 9500 and external systems, the connection setup of the PGA 9500 plays a crucial role. To establish a connection with the PGA, you need to follow a series of steps.

- First, ensure that the unit is powered on and connected to the network. The PGA can be integrated into various automation setups as it supports Ethernet communication.
- Once the unit is ready, you can proceed to configure the connection parameters. These settings determine how external systems can interact with the unit. The protocol used is OPC UA (Open Platform Communications Unified Architecture) which provides a standardized and secure method of communication. To initiate the connection external systems must make a connection request using the configured IP address and port number.
- Once connected bi-directional communication can take place. During the connection external systems can send commands to the PGA to control its operations. The PGA processes the commands and sends back responses, providing real-time information and notifications. It's important to handle connection errors and timeouts gracefully. The documentation provides details on error handling and troubleshooting tips to ensure a robust and stable connection. By following the connection setup guidelines in this documentation you'll be able to seamlessly integrate the PGA into your automation environment for efficient control and monitoring.

When you receive your new PGA 9500, the IP address of the unit is: http://192.168.1.69. To check if the connection can be established, it is possible to ping the unit. If the ping is successful, it should be possible to connect to the device's HMI user interface using a web browser (Google Chrome, Firefox, Edge...) with the address http://192.168.1.69:8080/webvisu.htm.

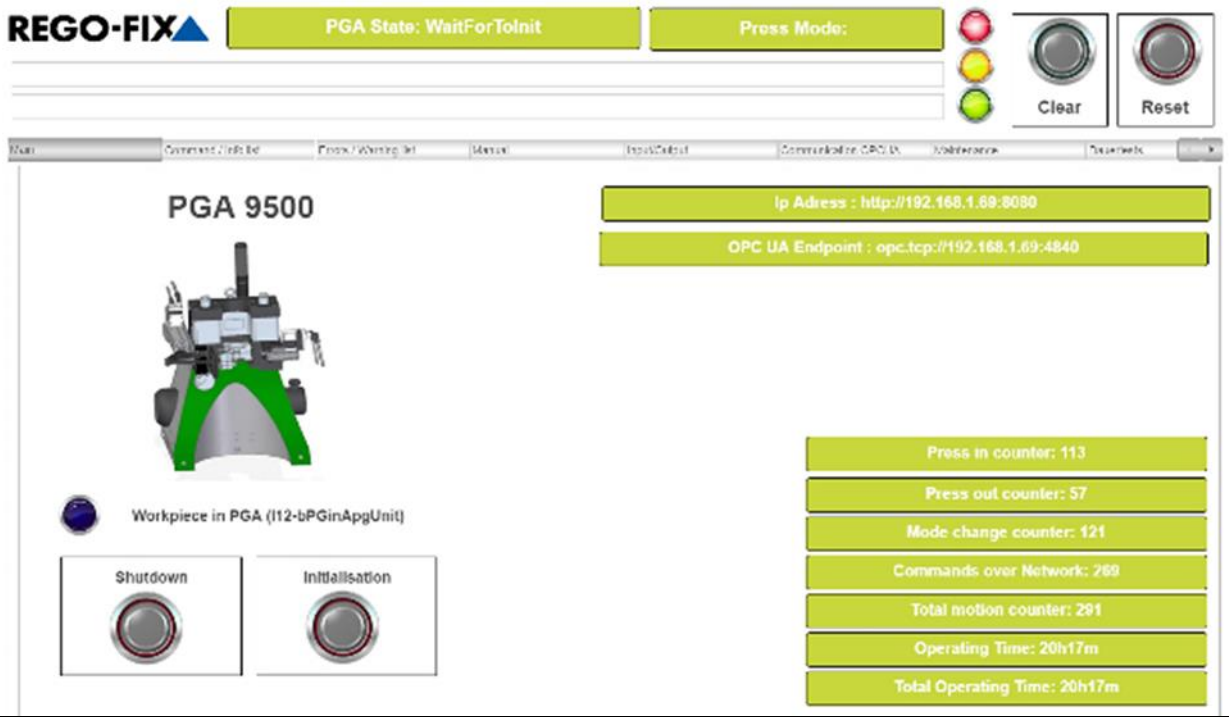

Fig.2: HMI startpage of the PGA visualization. When this page is displayed the connection to the PGA 9500 has been established successfully.

4.1. Check the connection

To check if an OPC UA communication can be established, an external client can be used named UaExpert ®. It is available on https://www.unified-automation.com/downloads/opc-ua-clients.html

4.2. OPC UA

- IP: 192.168.1.69
- HMI: http://192.168.1.69:8080/webvisu.htm
- OPC UA IP: opc.tcp://192.168.1.69:4840

Fig.3: Connection address on the PGA control cabinet

The QR-code on the sticker of the PGA control cabinet will guide you to the PGA information on the Github website (www.github.com).

4.3. UaExpert ®

UaExpert ® is a powerful software tool used for testing and troubleshooting of OPC UA servers and clients. With UaExpert Client users can easily connect to OPC UA servers, browse their address space, read and write data and monitor server performance. The tool supports various security mechanisms, including user authentication and encryption, ensuring secure communication between the client and server. With its intuitive user interface and a comprehensive set of features, UaExpert Client empowers users to efficiently validate OPC UA implementations, diagnose issues, and optimize the performance of their OPC UA devices.

4.3.1. UaExpert ® server

To connect to an OPC UA server, the user needs to press the “plus” sign and add a new server to the project.

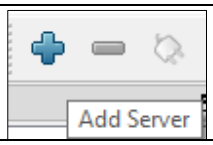

Fig.4: Add server

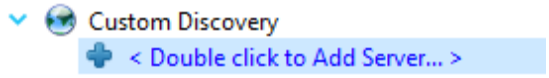

Fig.5: Custom Discovery

In the server selection-window you need to double-click on the “Add Server” layer. There are several subsections. It is important to choose the “Custom Discovery” section.

A window appears on the screen. There you need to enter the endpoint IP address which is given by initialization or you choose it by your own. The IP adresses can be found in the chapter “OPC UA”.

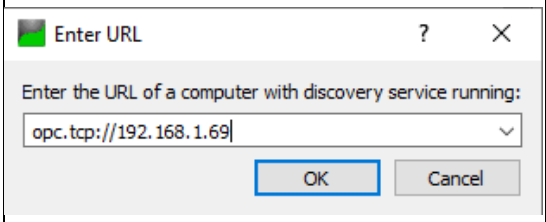

Fig.6: Endpoint IP address

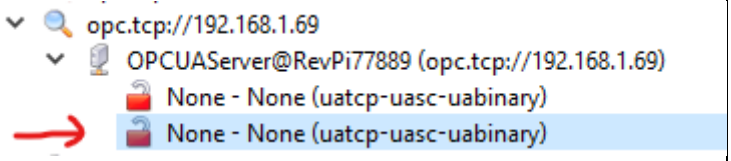

Fig.7: Server with name

When an OPC UA server is found a server icon with the server’s name and IP address appears below the IP address. If this doesn't pop up there could be a problem with the connection to the device. Check the network adaptor configurations and have a look in the troubleshooting section. Press OK to continue.

In the project there is now added a connection to the OPC server. Normally the connection establishes by itself. If it does not connect by itself press the connection button.

Fig.8: Connection to the project

When everything works fine the connection is established and you can see the same structure as in Fig.9 below. You can expand the folder root like below to the subfolder “PGA_Interface”. In this folder all variables to control the PGA device over a network connection are listed.

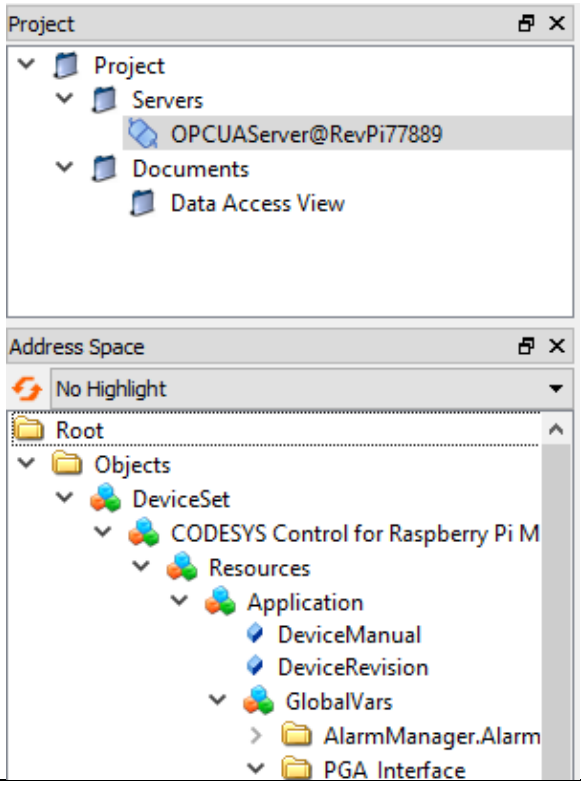

Fig.9: Project section

4.3.2. UaExpert ® hierarchy

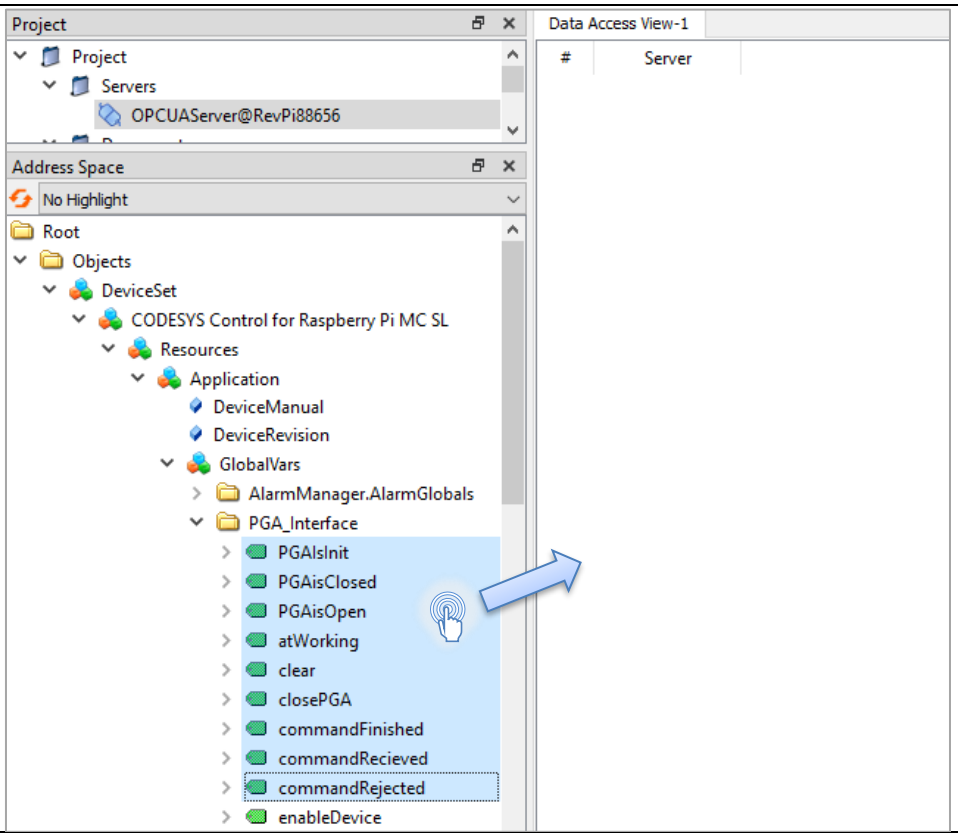

Fig.10: Project/UaExpert Root-List (PGA-Interface)

To visualize the data you can drag and drop the variables to the middle section of the window.

4.3.3. UaExpert ® example data

Over the data access view in UaExpert ® the variables can be manipulated and tested out. In the “status-code-raw” column the connection state can be seen. Variables can be set by clicking. Integers have numbers to change while booleans have a checkbox to set or unset.

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	PGAIsInit	false	Boolean	14:15:41.547	14:15:41.547	Good
2	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	PGAisClosed	false	Boolean	14:15:41.547	14:15:41.547	Good
3	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	PGAisOpen	false	Boolean	14:15:41.547	14:15:41.547	Good
4	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	atWorking	true	Boolean	14:15:41.547	14:15:41.547	Good
5	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	clear	false	Boolean	14:15:41.547	14:15:41.547	Good
6	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	closePGA	false	Boolean	14:15:41.547	14:15:41.547	Good
7	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	commandFinis...	false	Boolean	14:15:41.547	14:15:41.547	Good
8	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	commandRecie...	false	Boolean	14:15:41.547	14:15:41.547	Good
9	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	commandRejec...	false	Boolean	14:15:41.547	14:15:41.547	Good
10	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	enableDevice	false	Boolean	14:15:41.547	14:15:41.547	Good
11	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	error	0	UInt16	14:15:41.547	14:15:41.547	Good
12	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	execute	false	Boolean	14:15:41.547	14:15:41.547	Good
13	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	isEnabled	false	Boolean	14:15:41.547	14:15:41.547	Good
14	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	maintenance	false	Boolean	14:15:47.074	14:15:47.074	Good
15	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	manual	false	Boolean	14:15:47.074	14:15:47.074	Good
16	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChange	false	Boolean	14:15:47.074	14:15:47.074	Good
17	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChangeToIn	false	Boolean	14:15:47.074	14:15:47.074	Good
18	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChangeTo...	false	Boolean	14:15:47.074	14:15:47.074	Good
19	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChangeW...	false	Boolean	14:15:47.074	14:15:47.074	Good
20	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	notifications	0	UInt16	14:15:47.074	14:15:47.074	Good
21	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	openPGA	false	Boolean	14:15:47.074	14:15:47.074	Good
22	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	pause	false	Boolean	14:15:47.074	14:15:47.074	Good
23	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	paused	false	Boolean	14:15:47.074	14:15:47.074	Good

Fig.11: Example - data access view in UaExpert ®, PGA-Interface variables

5. Data model

The data model is a foundational concept in database management that defines how data is organized, structured and related within a system. It serves as a blueprint for data storage and manipulation, enabling efficient data retrieval and manipulation operations.

5.1. Permission

id	Description
r	read
w	write
rw	read + write

5.2. Variable/command list

Example for OPC UA connection string:

NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.Variable

Variable/command	Data-type	Permis-sion	Description
atWorking	Bool	r	The PGA will respond with true when the PGA is at work. If the PGA is not working the PGA will respond with false.
clear	Bool	w	If the device has had an error (e.g. safety circuit open), the PGA must be reactivated by toggling the variable to true for more than >1s or via the HMI interface. In addition, this bool resets all messages except errors but does not terminate any process and does not reset any states.
closePGA	Bool	w	With this Boolean variable the PGA door can be closed and will lock itself with the bolt. To start the closePGA cycle the execute boolean must be set to true. When the PGA has closed the door and locked it with the bolt, the PGA will respond to the PGAisClosed boolean with true. If the PGA has finished closing and locking the door the PGA responds to the commandFinished boolean with true. If the PGA has not closed and locked the door the PGA responds with false.
commandFinished	Bool	r	If the PGA has finished a command the PGA will respond with true. The state stays for one second active and then goes back to false. If the PGA respond with false the process is not finished yet.
commandReceived	Bool	r	The PGA will respond with true when the PGA has received a command. If the PGA has not received a command the PGA will respond with false.
commandRejected	Bool	r	If the PGA will respond with true a command was rejected. The true state is one second active and

Variable/command	Data-type	Permis-sion	Description
			goes back to false. If the PGA will respond with false everything is ok.
error	Uint	r	This read unsigned integer is used to check if there is an error in the PGA and to indicate it with a code. When the PGA has an error it will return a value >0, if there is no error the value remains 0.
enableDevice	Bool	w	This write boolean is used to enable the PGA. If the PGA is enabled the PGA will respond with true. If the PGA is not enabled the PGA will respond with false.
execute	Bool	w	If the PGA has executed a command the PGA responds with the variable “commandReceived” as true. If the PGA has not executed a command the PGA responds with the variable “commandReceived” as false.
isEnabled	Bool	r	This read boolean is used to check if the PGA is enabled. If the PGA is enabled the PGA will respond with true. If the PGA is not enabled the PGA responds with false
maintenance	Bool	r	The PGA will respond with true if the PGA is in maintenance mode. If the PGA is not in maintenance mode the PGA responds with false.
modeChange	Bool	w	This boolean can be used to change the mode of the PGA. To start the modeChange cycle, the execute boolean must be set to true. If the PGA has changed the mode the PGA will respond to the opposite press mode boolean (pressInMode/pressOutMode) with true. If the PGA has finished changing the mode the PGA responds to the commandFinished boolean with true. If the PGA has not changed the mode the PGA responds with false.
modeChangeToIn	Bool	w	This boolean can be used to change the mode of the PGA to the in mode if it isn't already in the in mode. To start the modeChangeToIn cycle the execute boolean must be set to true. If the PGA has changed the mode to in or was already in the in mode the PGA will respond to the pressInMode boolean with true. If the PGA has finished the mode change or was already in the in mode the PGA responds to the commandFinished boolean with true. If the PGA is not in the pressInMode the PGA responds with false.
modeChangeToOut	Bool	w	This boolean can be used to change the mode of the PGA to the out mode if it isn't already in the out mode. To start the modeChangeToOut cycle the execute boolean must be set to true. If the PGA has changed the mode to out or was already in the out

Variable/command	Data-type	Permis-sion	Description
			mode the PGA will respond to the pressOutMode boolean with true. If the PGA has finished the mode change or was already in the out mode the PGA responds to the commandFinished boolean with true. If the PGA is not in the pressOutMode the PGA responds with false.
modeChangeWith-Door	Bool	w	This boolean can be used to close the door by itself if it isn't already closed, change the mode and open the door all in one cycle. To start the modeChangeWithDoor cycle the execute boolean must be set to true. If the PGA has changed the mode the PGA will respond to the opposite pressing mode (pressInMode/pressOutMode) boolean with true. If the PGA has finished the mode change and opened the door the PGA responds to the commandFinished boolean with true. If the PGA has not changed the mode and has not opened the door the PGA responds with false.
notifications	Uint	r	If the PGA has a notification it returns a value >0, if there is no notification the value remains 0.
openPGA	Bool	w	With this boolean the PGA door can be unlocked and opened by itself. To start the openPGA cycle the execute boolean must be set to true. When the PGA has opened the door and unlocked the bolt the PGA will respond to the PGAisOpen boolean with true. If the PGA has finished unlocking and opening the door the PGA responds to the commandFinished boolean with true. If the PGA has not unlocked and opened the door the PGA responds with false.
paused	Bool	r	If the PGA is paused it responds with true. If the PGA is not paused it responds with false.
PGinAPGunit	Bool	r	This boolean is used to check if the PGA has a toolholder in the APG. If the PGA has a toolholder in the APG the PGA will respond with true. If the PGA has no toolholder in the APG the PGA will respond with false.
PGAisClosed	Bool	r	If the door is closed and locked with the bolt the PGA will respond with true. If the door is not closed the PGA will respond with false.
PGAIsInit	Bool	r	If the PGA is initialized, in the press-in mode and the door is open, the PGA will respond with true. If the PGA is in not initialized and has no open PGA signal, it responds with false.
PGAisOpen	Bool	r	If the door is unlocked and open the PGA will respond with true. If the door is closed the PGA will respond with false.

Variable/command	Data-type	Permis-sion	Description
pressFinish	Bool	r	If the pressing process is finished the PGA responds with true. The state remains active for one second and then changes back to false. The process is not finished if the PGA returns to false.
readyForCommands	Bool	r	If the PGA is ready to receive commands the PGA will respond with true. If the PGA is not ready for commands the PGA will respond with false.
reset	Bool	w	This boolean is used to reset the PGA by toggling the variable to true for more than >1s. All states and errors/warnings are reset.
shutdown	Bool	w	This boolean is used to shut down the PGA controller.
spaceFree*	Bool	w	This is a very important variable. It is used to check whether the PGA door and bolt can be moved, i.e. the space in front of the door is free. If this boolean is not active, the door will not move. The PGA will complete the pressing cycle and will pause all commands.
pressInComplete	Bool	w	This boolean variable is used to close the door by itself if it isn't already closed, press in the collet and open the door all in one cycle. To start the pressInComplete cycle, the execute boolean must be set to true. If the PGA has fully pressed in the collet, the PGA will respond to the pressFinish boolean with true. If the PGA has not fully pressed in the collet the PGA responds with false. If the PGA has finished the hole cycle the PGA responds to the commandFinished boolean with true. If the PGA has not pressed the collet and has not opened the door it responds with false.
pressInIncomplete	Bool	w	This boolean is used to close the door by itself if it isn't already closed, press in the collet all in one cycle, but not to open the door after pressing in. The execute boolean must be set to true to start the PressInIncomplete cycle. If the PGA has fully pressed in the collet the PGA will respond to the pressFinish boolean with true. If the PGA has not fully pressed in the collet the PGA will respond with false. If the PGA has finished the hole cycle the PGA responds to the commandFinished boolean with true. If the PGA has not pressed in the collet it responds with false.
pressOutComplete	Bool	w	This boolean is used to close the door by itself if it isn't already closed, press out the collet and open the door all in one cycle. To start the pressOutComplete cycle the execute boolean must be set to true. If the PGA has fully pressed out the collet the PGA

Variable/command	Data-type	Permis-sion	Description
			will respond to the pressFinish boolean with true. If the PGA has not fully pressed out the collet the PGA responds with false. If the PGA has finished the hole cycle the PGA responds to the command-Finished boolean with true. If the PGA has not pressed out the collet and has not opened the door it responds with false.
pressOutIncomplete	Bool	w	This boolean is used to close the door by itself if it isn't already closed, press out the collet all in one cycle, but not to open the door after pressing out. The execute boolean must be set to true to start the PressOutIncomplete cycle. If the PGA has fully pressed out the collet the PGA will respond to the pressFinish boolean with true. If the PGA has not fully pressed out the collet, the PGA will respond with false. If the PGA has finished the hole cycle the PGA responds to the commandFinished boolean with true. If the PGA has not pressed out the collet it responds with false.

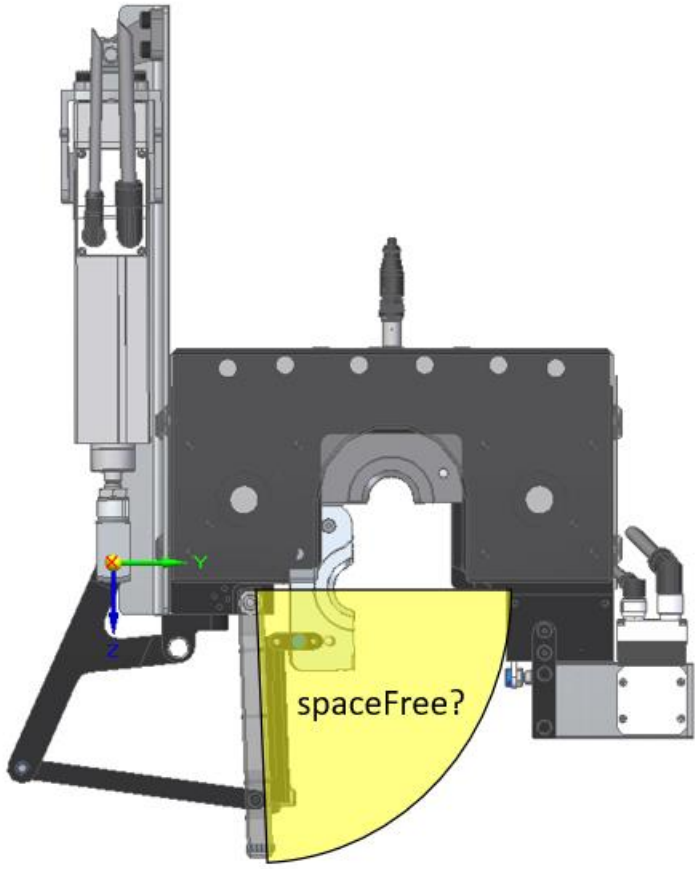

Fig.10: Additional information for spaceFree variable*

6. Network procedures

The PGA 9500 offers various features and functions that can be accessed and controlled via a network interface. You can enable and disable the PGA over the network, retrieve its status, send commands, handle errors, change operating modes, manage collets and holders with a robot and control the PGA in detail. Each procedure is accompanied by diagrams and explanations to facilitate a clear understanding of the API functionality.
The PGA 9500 unit interacts with the API through read and write boolean variables. When working with the PGA unit, users can access various read boolean variables to retrieve specific status information. For example, users can query the operational mode read boolean variable to determine the current mode in which the PGA is operating. To initiate actions or modify settings on the PGA unit, users employ write boolean variables. Before executing a command, users need to select the corresponding command boolean variable that represents the desired action. This selection specifies the operation to be performed on the device. Once the command boolean variable is set, users trigger the execute boolean variable, which initiates the execution of the command. This two-step process ensures that commands are intentionally initiated and helps prevent accidental actions.
During the execution of a process the PGA provides feedback on the status of the operation. This feedback is typically reflected in other boolean variables, allowing users to monitor the progress of their actions. For example, a start boolean variable may indicate that a process has begun while a success boolean variable may confirm that the process was completed successfully. Similarly, if a process fails an error boolean variable may be set to indicate the failure. By monitoring these status variables, users can track the progress of their operations and respond accordingly. By understanding the interaction between read and write boolean variables users can effectively communicate with the PGA unit through the API.
Additionally, it is important to note that only one command can be active at a time when interacting with the PGA through the API. This means that before initiating a new command the previous command must be completed or canceled. By enforcing this restriction, the PGA ensures that commands are executed in a controlled manner, preventing conflicts or overlapping actions. This design ensures the integrity of the device's operations and avoids any potential issues that may arise from concurrent command execution. To manage the execution of commands effectively users should carefully track the status of the ongoing commands and ensure that they are completed or canceled before initiating new ones. This can be achieved by monitoring the relevant boolean variables associated with the command's execution status.

6.1. Enable PGA

Once the device is ready, users can confidently send commands to control and configure the PGA according to their specific requirements.

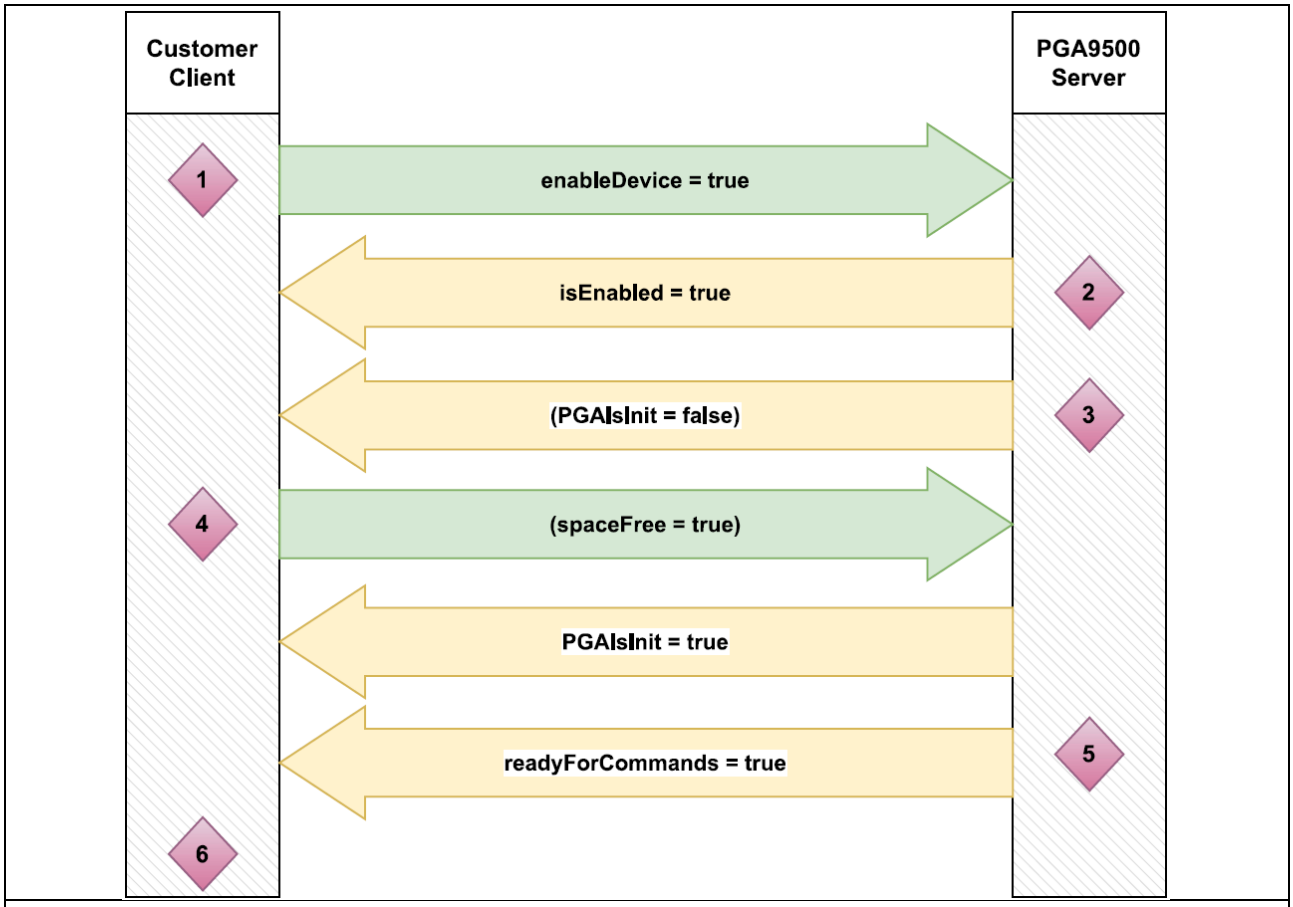

Fig.12: Flowchart of the enabling process (Initialization)

1) The device is powered on but is in the disabled state. The OPC UA connection is established, and the device is ready to receive commands. The device is not ready to execute commands. The enableDevice boolean variable is set to false and will be set to true to enable the device. In order to initialize the PGA, it is important that there is no toolholder placed in the PGA and the PGinAPGunit variable is false.

   → Initialization is impossible with a toolholder inside the PGA.

2) After setting the enableDevice boolean variable to true, the device will start the enabling process and will set the isEnabled boolean variable to true. This indicates that the device is enabled.

3) After rebooting or resetting the device, the device needs to do an initialization before it is ready to receive commands.

4) To initialize the PGA automatically, the robot or handling system must signalize with the variable spaceFree that the door space is free. This is because the door/bolt drives and their sensors also adjust and move during the initialization process.

5) The PGA is enabled and initialized and sets the readyForCommands variable to true.

6) When this is done by the first time, the device is initialized in the "press In" mode and the "press In" mode is set to true. The device is now ready to receive commands and execute them.

6.2. Disable PGA

The process of disabling the PGA unit over a network interface ensures the safe disconnection and termination of the device functionalities. By following a systematic approach users can initiate the disabling procedure, effectively halting command execution and severing the connection between the PGA and external systems.

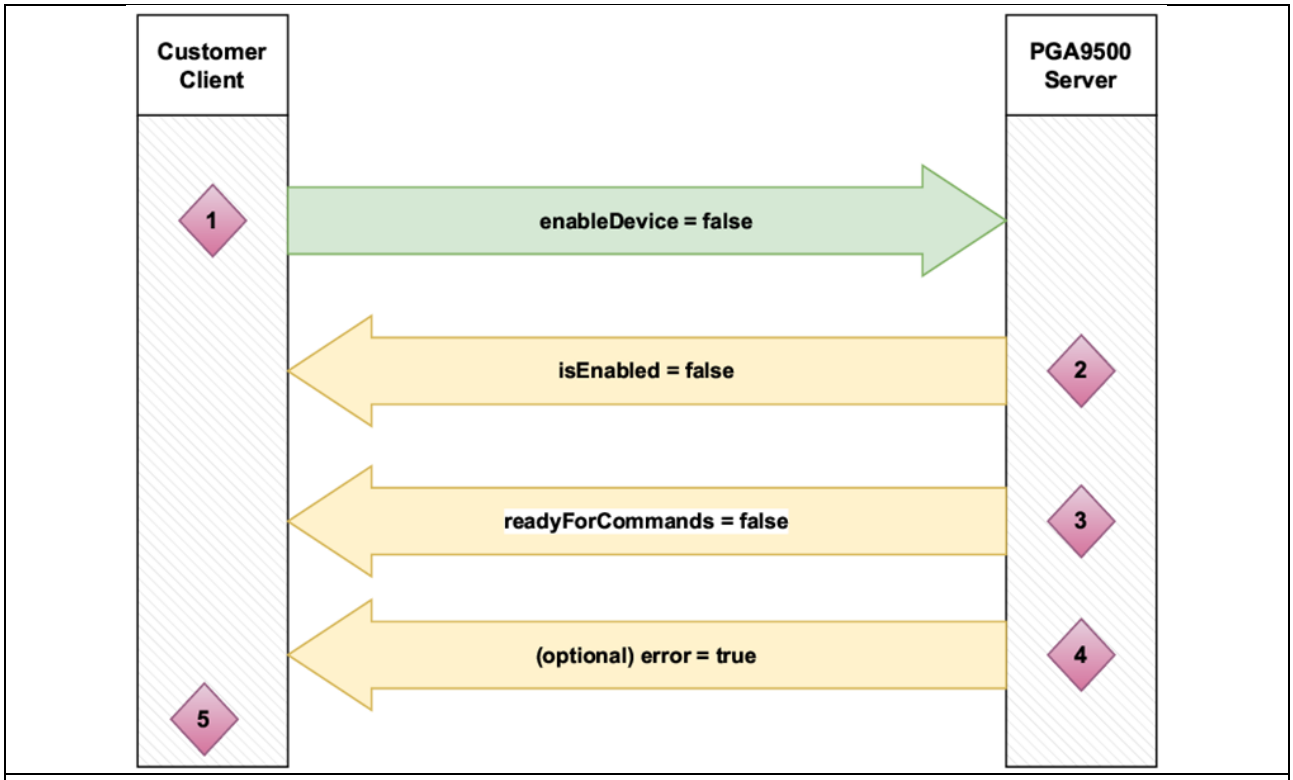

Fig.13: Flowchart of the disabling process

1) The unit is enabled and ready to be disabled. The OPC UA connection is established, and the device is ready to receive commands. The device is ready to execute commands. The enableDevice boolean variable is set to true and will be set to false to disable the unit.

2) After setting the enableDevice boolean variable to false the device will start the disabling process and will set the isEnabled boolean variable to false. This indicates that the device is disabled and the initialization process is finished.

3) In addition to the isEnable variable the unit will also set the readyForCommands boolean variable to false. This indicates that the device is not ready to receive commands. The device does not receive commands and execute them.

4) If something goes wrong during the "disable" process the device will also set the error boolean variable to true. This indicates that the unit is in an error state and the process is not complete. This may also occur if the device was disabled while a process was running. The exact error message can be found in the error table.

5) The unit is now in the disabled and not ready state.

6.3. Get state of PGA

Various status indicators can be read from the device including maintenance status, safety information, error conditions, press-in and press-out mode, door state, bolt state, and toolholder presence within the unit. By following the instructions outlined in this guide, users can effectively access and interpret these status parameters, enabling them to monitor the device condition, ensure safety compliance, and make informed decisions regarding maintenance, process control, and toolholder handling.

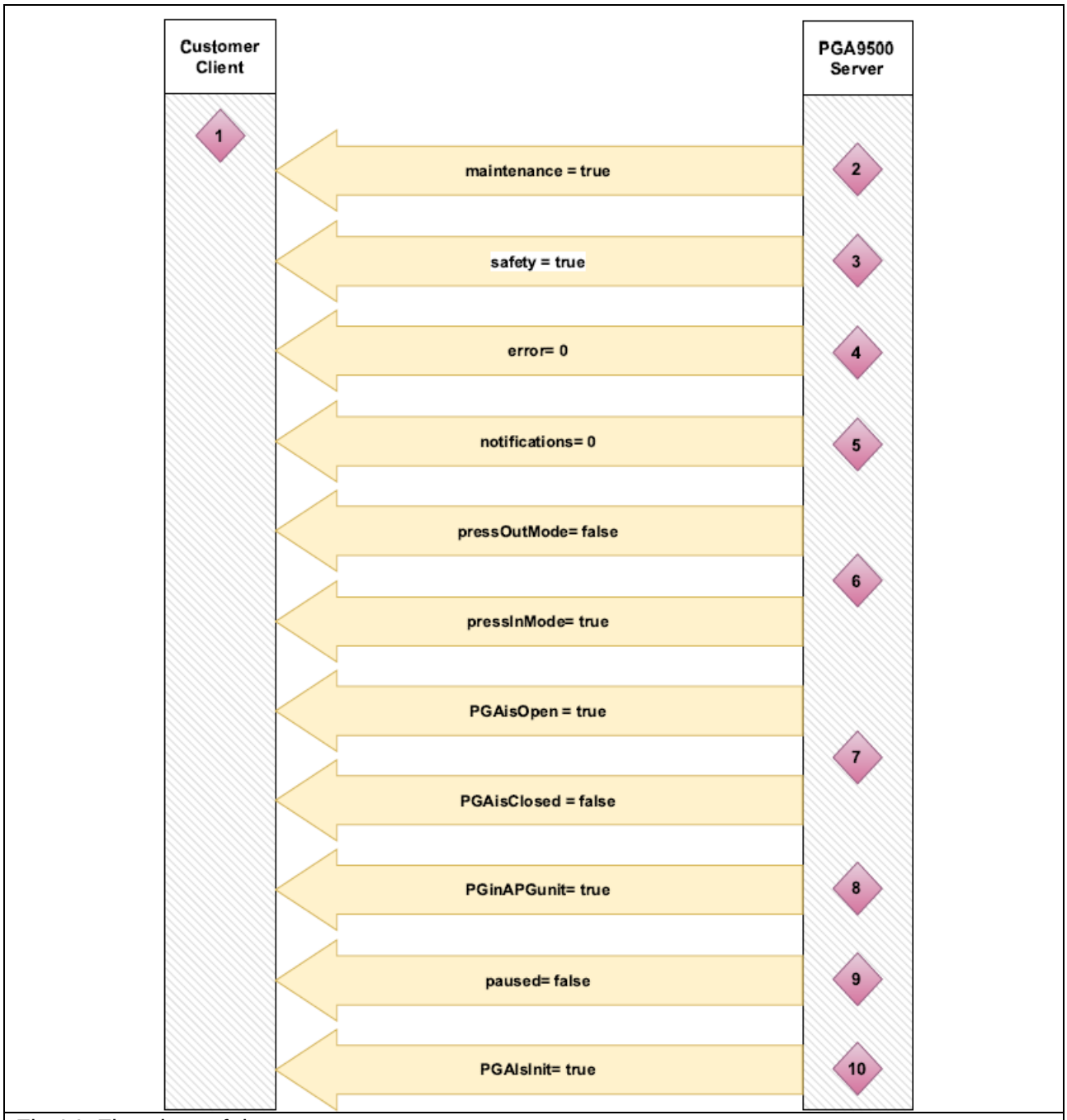

Fig.14: Flowchart of the states

1) All states can be read in any state or at any time when the unit is online, powered on and the OPC UA connection is established. The device must not be enabled to read the state. The device is in the disabled state. The device is not ready to execute commands. The boolean variable enableDevice is set to false and will be set to true to enable the device.

2) The maintenance boolean indicates, with a positive true, that the device has a maintenance problem that should be rectified. This could be the oil or a seal that needs to be changed.

The exact maintenance message and its description for the associated code can be found in the maintenance table.

3) The safety boolean is true when the safety circuit is closed. This means that the external safety chain is closed and no emergency stop button is pressed. This circuit can be opened without an error message and if it was opened, it must be cleared with the clear boolean to continue.

4) The unsigned integer error is >0 if the unit is in an error state and indicates a specific error with the associated error code. The exact error and its description for the associated code can be found in the error table.

5) The unsigned integer notifications is >0 if the unit has a notification with the associated notification code. The exact notification and its description for the associated code can be found in the notification table.

6) The pressOutMode boolean indicates the mode of the unit. If the boolean is true the device is in press out mode. If the boolean is false the device is in the press in mode. If the pressInMode boolean is true the unit is in the press in mode and the pressOutMode boolean is false.

7) The PGAisClosed boolean indicates the status of the door and bolt. If the boolean is true the door is closed and locked with the bolt. If the boolean is false the bolt and door are open and the boolean PGAisOpen is true.

8) A special indicator is the toolholderInPGA boolean. This indicates when a tool is inside the PGA. This can be useful to avoid collisions after a reset or initialization. If the variable is true a tool is in the PGA. If the variable is false there is no tool in the PGA and the space is free.

9) The paused boolean indicates whether the unit has been paused or not. If the boolean is true the device is paused and, if started, completes the pressing, then stops and pauses all commands. If the boolean is false the device operates normally and is not paused.

10) The PGAisInit boolean indicates whether the PGA is being initialized. If the boolean is true the device is initialized. If the boolean is false it is not initialized and not ready for commands. The first time it is enabled with spaceFree, it will initialize itself.

6.4. Send commands to PGA

This comprehensive guide provides a detailed explanation of the step-by-step sequence required to send commands effectively, covering optional elements in the process. The guide encompasses 12 essential steps, including the selection of a command boolean, executing the command and receiving confirmation. By following these instructions, users can navigate the communication process seamlessly, ensuring accurate command transmission and facilitating efficient control of the PGA unit. Understanding the command structure and adhering to the prescribed sequence empowers users to interact with the PGA 9500 effectively, unlocking its capabilities for a wide range of applications.

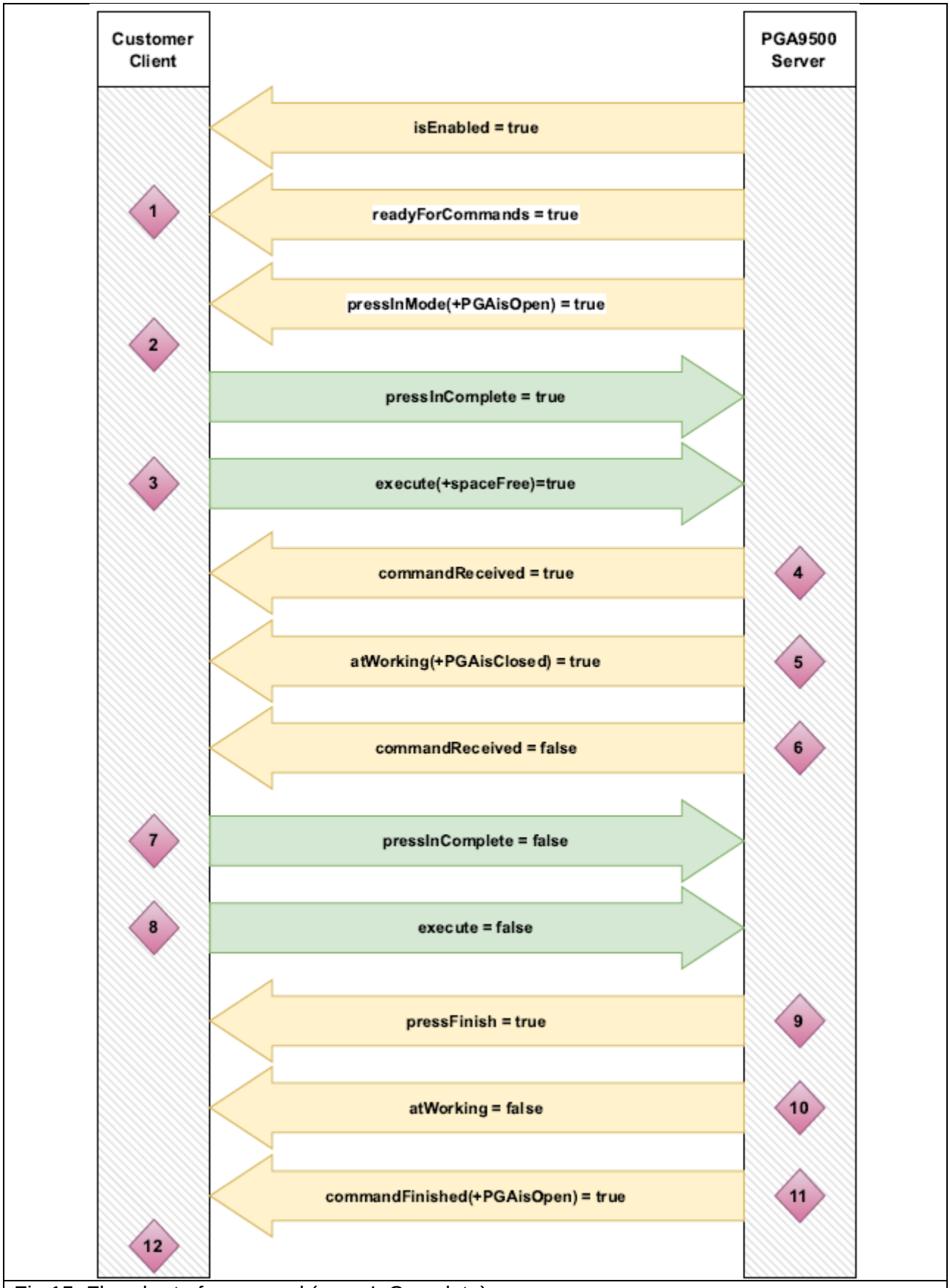

Fig.15: Flowchart of command (pressInComplete)

1) Before a command can be executed or set, the device must be enabled like described in the enable section. The OPC UA connection is established, and the device is ready to receive commands. Important is that the isWorking variable is on false and the readyForCommands variable is on true.

2) When the preconditions are fulfilled any command can be set. The commands can be true like in the example above pressInComplete. There could only be set one command at the same time. When there is more than one command set the device cannot manage them and will rise an error. So, it is important to set only one command at the same time. The list of the variables/commands is listed in the table above.

3) For the example pressInComplete the PGA must be in the pressInMode (for details to set the right mode see the variable modeChange). If one command is selected, the execute variable can be set to true. This will execute the command. If the PGAisOpen variable is true its necessary to set the spaceFree variable to true, so the PGA is allowed to close the door. If it's already closed the spaceFree variable is not relevant.

4) After that command is executed the PGA register the command and when everything is fine to execute, the commandRecieved value is set to true. That indicates that the PGA accepts the command and will execute it. Additionally, the variable readyForCommands is set to false. This indicates that the PGA is not ready to receive another command.

5) With the atWorking variable the PGA indicates that the PGA is in process and works on the command that is set. In the example above, the door of the device will close and the PGAisClosed variable will be set to true, if it wasn't already.

6) After 1-2 seconds the commandRecieved variable is set to true and the PGA turns the commandRecieved variable to false. That is only for reseting the state of the commandRecieved variable.

7) While the command is executing and the commandRecieved variable confirms that the command is working, it is possible to set the command to false.

8) The same applies for the execute variable.

9) When the PGA is finished with the pressing of the collet itself the PGA sets the pressFinish variable to true.

10) When the PGA is finished with the hole command the isWorking variable is set to false and indicates that the PGA is no more working on a command.

11) In the example the PGAisOpen will be set to true after the door has opened automatically (not in pressIncomplete). An additional indicator is the commandFinished variable. When this variable turns to true the command is safely and correct finished. The process is done and has no errors or issues. This variable changes back to false after a few seconds.

12) When everything works fine the readyForCommands variable is set to true and the PGA is ready to receive commands again. When some errors appear while executing a command the PGA indicates this with the error variable as a code and the details are described in the error table below.

6.5. Mode change

This article explores the mode change functionality of the PGA unit over a network interface. The focus is on the two available modes: Press In Mode and Press Out Mode. Users can switch between these two modes by employing a similar command structure as described in the previous chapters. Three different methods for mode change are discussed:

1. transitioning directly to the opposite mode,
2. transitioning to a specific mode (e.g., Press In Mode)
3. transitioning directly to the opposite mode while also closing the door.

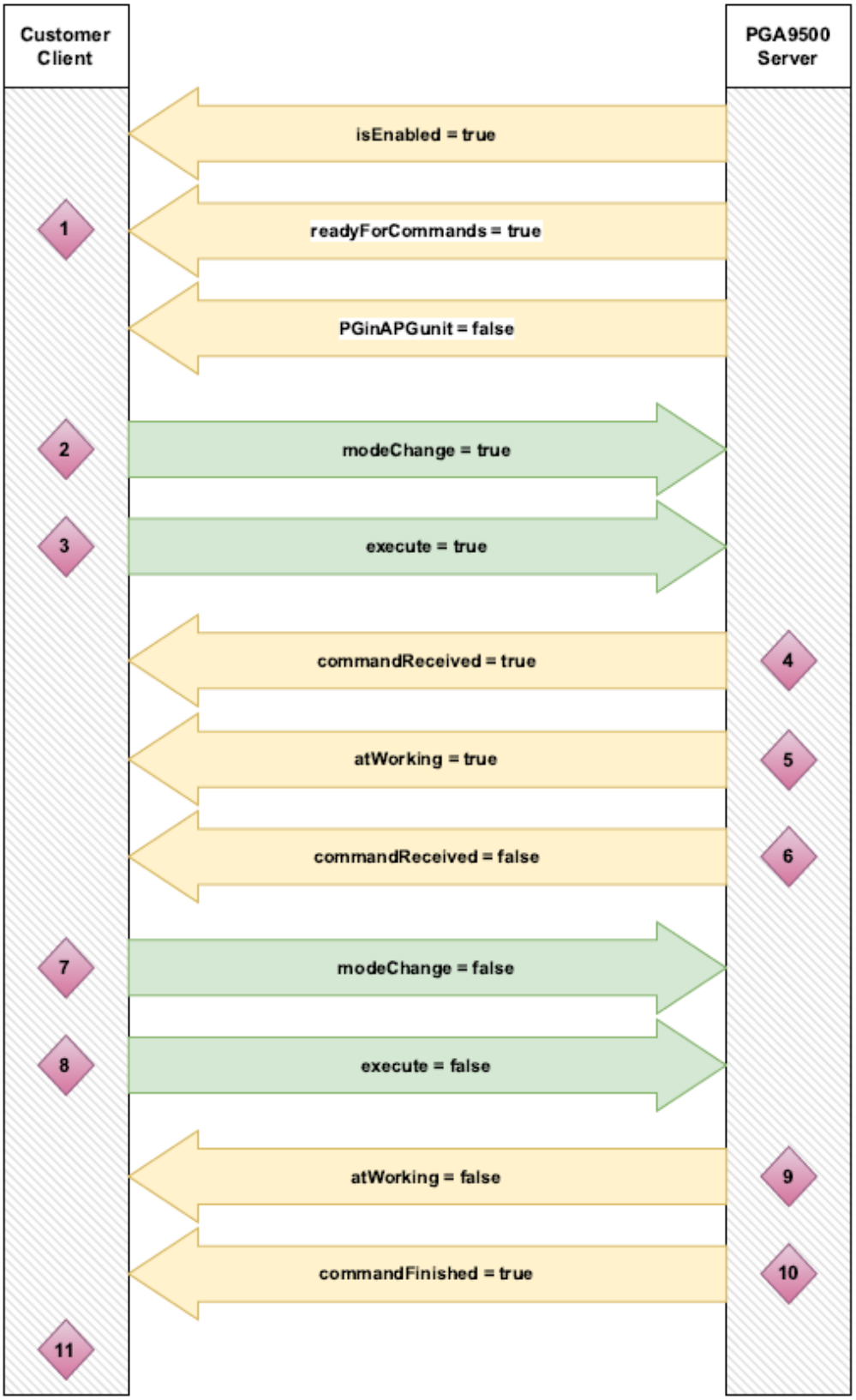

Fig.16: Flowchart of the mode change

1) Before a mode change can be executed or set the unit must be enabled like described in the enable section. The OPC UA connection is established and the device is ready to receive commands. Important is that the atWorking variable is on false, the readyForCommands variable is on true and no toolholder is placed in the PGA. It's impossible to change the mode with a toolholder inside.

2) If the preconditions are fulfilled any mode change can be set. The mode changes can be like in the picture above pressInMode to true. There could only be set one mode change at the same time. When there are more than one mode change set the device cannot manage them and will rise an error. So, it is important to set only one mode change at the same time. The list of the mode changes is in the table above in the mode change section.

3) If one mode change is selected the execute variable can be set to true. This will execute the mode change.

4) After a mode change is executed the PGA registers this mode change and when everything is fine will execute it. The commandRecieved value is set to true. This indicates that the PGA accepts the mode change and will execute it. Additionally, the variable readyForCommands is set to false. This indicates that the PGA is not ready to receive commands.

5) With the atWorking variable the PGA indicates that the PGA is in process and works on a mode change that is set.

6) After 1-2 seconds the commandRecieved variable is set to true, the PGA turns the commandRecieved variable to false. This is only for resetting the state of the commandRecieved variable.

7) While the mode change is being executed and the commandReceived variable has confirmed that the mode change is working it is possible to set the mode change to false.

8) The same applies for the execute variable.

9) If the PGA is finished with the given mode change the isWorking variable is set to false and indicates that the PGA is no more working on a mode change.

10) An additional indicator is the commandFinished variable. If this variable turns to true the mode change is safely and correct finished. The process is done and has no errors or issues. This variable changes back to false after a few seconds

11) If everything works fine the readyForCommands variable is set to true and the PGA is ready to receive mode changes again. The mode change has worked successful and the PGA is ready for the next mode change. If some errors appear while executing a mode change the PGA indicates this with the error variable and a specific code.

6.6. Detailed PGA control

An external device, such as a robot or a handling system starts the process by inserting a PG toolholder into the PGA 9500. Once the toolholder is inserted the space in front of the door is free and the PGA door is closed, the PGA is signaling to proceed with the subsequent steps. Upon detecting the closed PGA it allows the robot to release the toolholder. During this phase the automation system no longer needs to actively hold the toolholder as the PGA takes over the responsibility of keeping it in place. Before the door reopens the external automation system must regain control and ensure the toolholder is firmly held and the space in front of the door is free again. This step is essential to maintain stability and prevent any unwanted movement or dislodging of the toolholder during the subsequent operation. With the external system holding the toolholder and signaling a free space, the door of the PGA opens, granting access to the inserted toolholder. The open door allows the robot to safely extract the pressed-in toolholder from the PGA. Care should be taken to ensure proper alignment and secure handling during this step to avoid any damage of the toolholder or the device. Once the extraction process is completed the extracted toolholder is available for reuse in subsequent operations. The PGA is now ready for the next tool insertion or extraction cycle. It is important to note that this process cycle is applicable to PG toolholders of all sizes and it applies to both tool insertion and extraction operations. The synchronized coordination between the PGA unit and the external automation system ensures efficient and reliable tool handling throughout the operation.

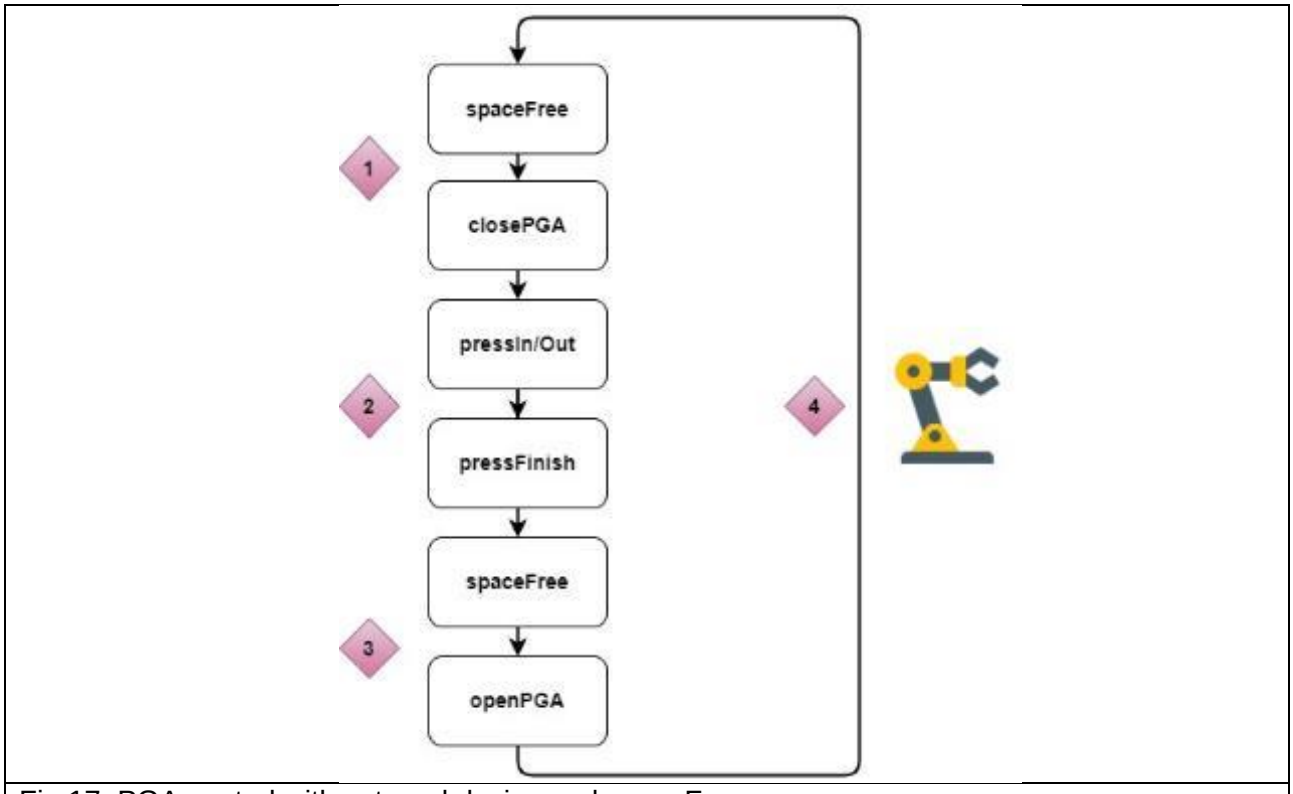

Fig.17: PGA control with external device and spaceFree

1) The PG toolholder is inserted into the PGA and the PGA is enabled, initialized and ready to receive any commands. The robot or handling system is not in the motion space of the door. The PGA closes automatically or can be closed over network ((closePGA+)pressXComplete, (closePGA+)pressXIncomplete).

2) The PGA can start with the pressing procedure. It is the same procedure for the press in or for the press out mode. The PGA indicates when the pressing procedure is finished.

3) The robot or handling system is not in the motion space of the door. The PGA opens automatically or can be opened over network (pressXComplete, pressXIncomplete+openPGA). For the automation system around the PGA, it is important to hold the PG toolholder otherwise the PG toolholder could fall out of the PGA.

4) When this process is finished, the robot can interact with the robot or handling system and the PGA is ready for the next pressing step.

6.7. Change toolholder with robot feedback

One of the remarkable features of the PGA 9500 unit is its ability to provide real-time feedback on the status of the inserted toolholder. This ensures effective monitoring and enables users to stay informed about the presence or absence of a tool inside the PGA. To initiate the toolholder status monitoring process a robot is utilized to insert a toolholder into the PGA. The device is equipped with a sensor that actively detects the presence of a toolholder. As the robot places the toolholder into the PGA the sensor feedback becomes active, indicating that the toolholder is securely inserted and ready for operation. Once the toolholder is inserted the sensor feedback remains active, providing continuous monitoring of the toolholders presence within the PGA. This real-time status information can be accessed through the API allowing users to retrieve the toolholder status as needed. When the robot removes the toolholder from the PGA the sensor feedback is deactivated, indicating that the toolholder is no longer present in the unit. This change in status can be monitored through the API providing prompt updates to the external automation system.

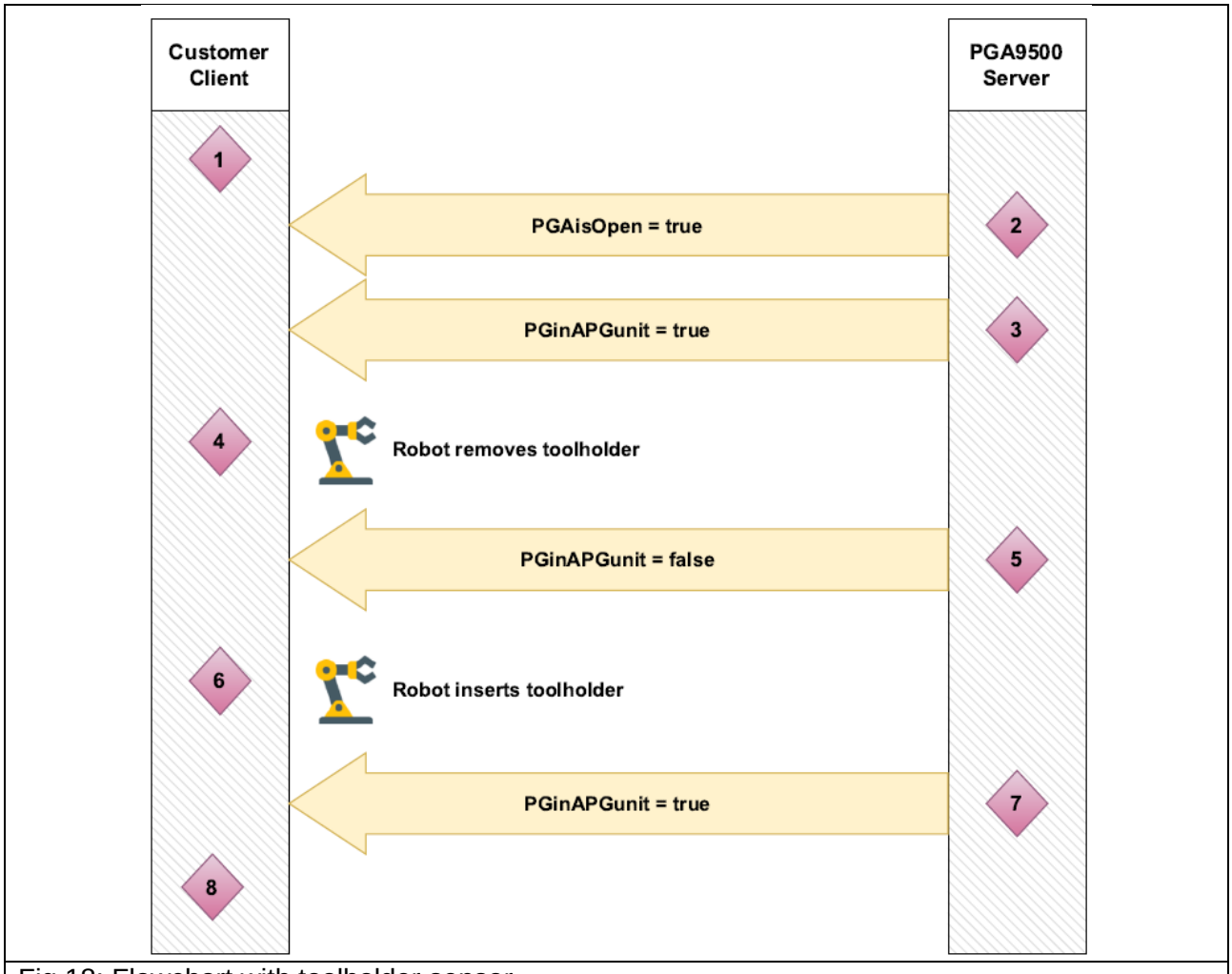

Fig.18: Flowchart with toolholder sensor

1. The unit is enabled and ready to receive any command. A toolholder is placed into the PGA and the process of pressing in or out is completed.
2. The PGAisOpen variable is on true, that means the door is open and the robot can interact with the PGA. In addition, the spaceFree variable could be set to false so that it is impossible for the door to move and the robot can safely place a new toolholder into the PGA.
3. On the OPC UA interface the PGinAPGunit is true because a toolholder is in the PGA and can be grabbed by the robot.
4. The robot can take the toolholder with a linear line interpolation out of the PGA.
5. When the toolholder is removed the PGinAPGunit variable turns to false and is signaling that no toolholder is in the unit.
6. The robot can change the toolholder and place it into the PGA.
7. As soon as the toolholder is back in the PGA the variable turns to true and is signaling that the unit is ready for a pressing process.
8. The device is ready to receive new commands and start any process with the new toolholder. Remember that the robot should only release the toolholder when the PGA is closed.

6.8. Safety handling

Another notable feature of the PGA is the ability to handle an interruption in the safety circuit. The safety concept ensures a safe stop of all relevant systems without having to completely leave the process. This ensures effective monitoring and allows the user to resume operation from the same point in case of an interruption during operation. The PGA provides an externally switchable safety circuit that is continuously monitored. The user can access the status of the safety circuit via the API. The user is also informed on further steps via notification signals. During a pressing process the PGA monitors whether the process has been completed. If it has been aborted by the safety during the process the device recognizes that it needs to perform that specific process step again. This ensures that the process can continue after an interruption without any loss of information and that the pressing process is fully completed. All the users must do is to confirm the interruption to continue the process.

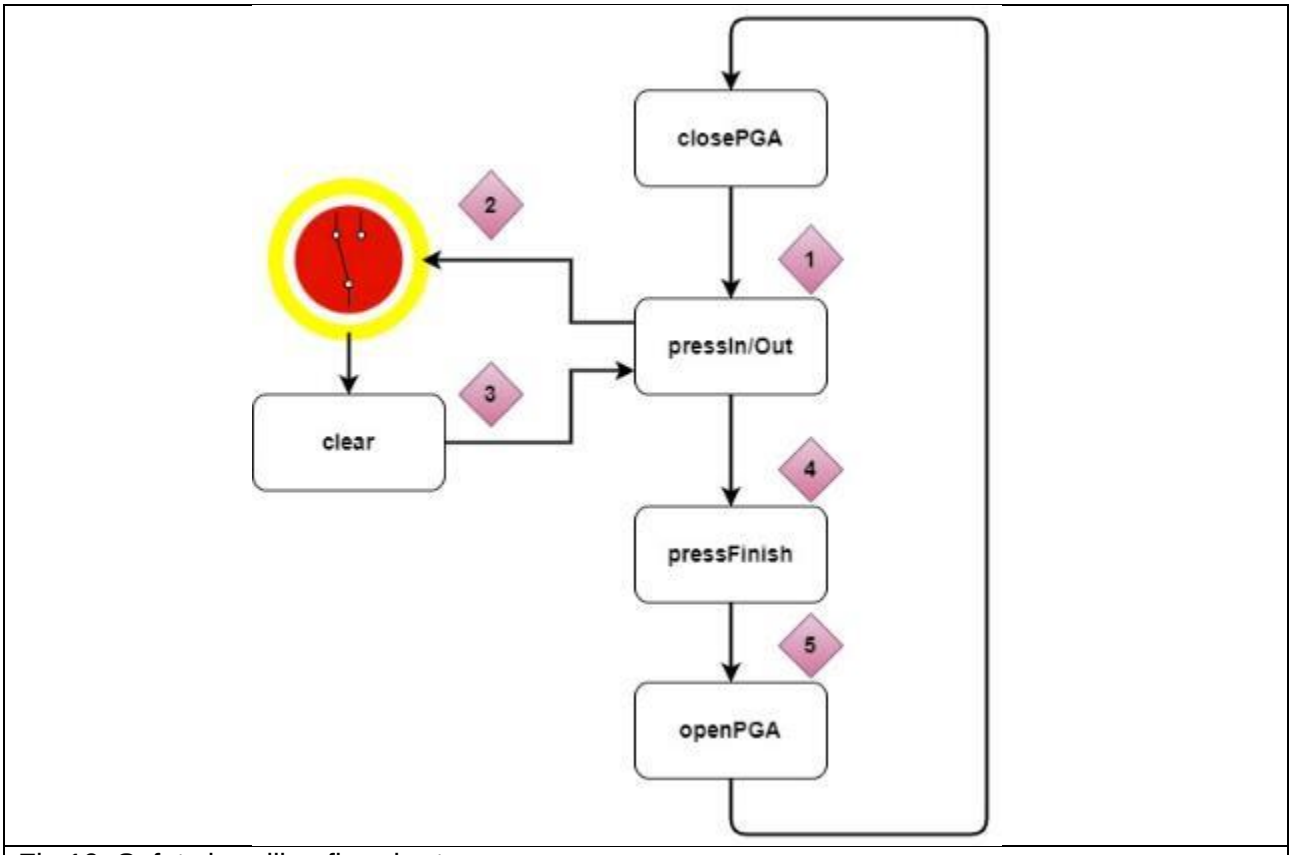

Fig.19: Safety handling flowchart

1. The PG toolholder is inserted into the PGA and the PGA is enabled, initialized and ready to receive any command. The robot or handling system is not in the motion space of the door. The PGA is closed and receives the command to start a pressing process.
2. The PGA starts the pressing process. The procedure is the same for press in and press out. This is stopped by interrupting the safety circuit before completion and receiving the true signal from the pressFinish variable. This does detect that the collet is not completely pressed in/out. The safety variable is set to false.
3. After leaving the safety area and closing the safety circuit again the safety variable is set to true. This must be confirmed by setting the variable to true for >1s to continue the process. The pressing cycle has not been completed. The PGA will move to the initialization position and then finish the process. If the safety circuit was interrupted after the PGA has been closed but the pressing process has not yet been started, the door opens and closes again when it is restarted.
4. The pressing process is fully completed and the pressFinish variable is set to true.
5. When the safety circuit was triggered the power supply of the door drives will be interrupted too. After closing the safety circuit a few seconds are required to be able to receive commands again. The process of opening/closing the door is paused and will be resumed automatically when the door drives are ready again. The notification variable reports this event with the corresponding code.

7. Error handling and notifications

7.1. Error handling example

This section describes how to handle an error over the network using the PGA interface. An example of an error event is described. With this step-by-step description it is possible to troubleshoot an error event and, if possible, complete the failed process. It is important to note that the safety circuit needs to be open when inspecting close to the PGA.

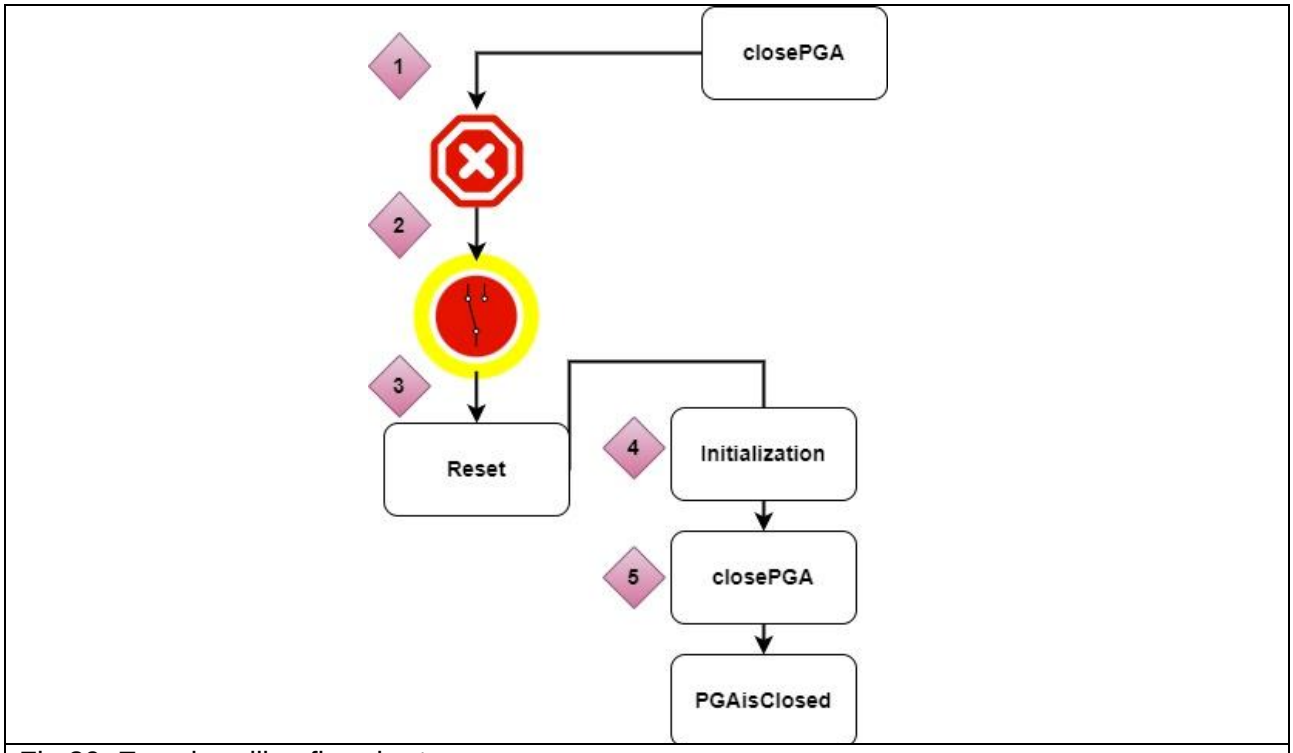

Fig.20: Error handling flowchart

1. The PGA is enabled, initialized and ready to receive any command. The PGA receives the command to close the door. An error has occurred in the meantime. The door cannot be closed. The exact error is indicated by a code on the error variable. The exact description of the error code can be found in the error table.
2. Disconnect the safety circuit after a quick check of the PGA. If the door was blocked the drive is now disengaged and the force on the door is turned off. The door can be moved now, and the blockage removed.
3. After leaving the safety area and closing the safety circuit again, the safety variable is set to true. The reset variable must be set to true for >1s. It is also recommended to reset all user commands and signals as well. The error code should be reset and if fixed should no longer appear.
4. After reset, the PGA must be initialized. The enableDevice variable is set to true and the door movement space is reported free by setting the spaceFree variable to true, which starts the automatic initialization. Once initialization is completed the device is ready to receive and execute commands.
5. The PGA is enabled, initialized and ready to receive any commands, the space of the door is free and the command to close the PGA is sent again. The door is closed and locked.

7.2. Error codes

Error Code	Name	Description	Solution
0	No error	No error	No error
1	Door cannot be closed	Door close timeout error	Possible error on the motor or the power supply. There could also be something between the door and the PGA.
2	Door cannot be opened	Door open timeout error	Possible error on the motor or the power supply. There could also be something between the door and the PGA.
3	Unknown command	The command was not recognized by the PGA.	An unknown command was used, or the command was misspelled. Check the command and try again.
4	Error during booting phase	The PGA is not ready to receive any command.	Check the PGA and try again (reboot the unit).
5	Mode cannot be changed	The mode cannot be changed. Tool inside the PGA	Check the PGA, remove the toolholder or update the calibration of the toolholder sensor
6	The PGA cannot press in	Signal problem or PGA error.	Check the connection and maybe the wiring to the PGA.
7	The PGA cannot press out	Signal problem or PGA error.	Check the connection and maybe the wiring to the PGA.
8	Bolt cannot be closed	Bolt close timeout error	There must be an error on the motor or the power supply. There could also be something between the bolt and the PGA.
9	Bolt cannot be opened	Bolt open timeout error	There must be an error on the motor or the power supply. There could also be something between the bolt and the PGA.
10	Tool inside the PGA while initialization	Impossible to do initialization with a toolholder inside the PGA	Check the PGA, remove the toolholder or update the calibration of the toolholder sensor
11	Bolt drive sensor problem	Bolt sensor is opened and closed at the same time	There must be an error with the sensors, or the settings of the sensors/drives are wrong.
12	Door drive sensor problem	Door sensor is opened and closed at the same time	There must be an error with the sensors, or the settings of the sensors/drives are wrong.

7.3. Notification codes

Maintenance Code	Name	Description	Solution
0	No error	No error	No error
1	The commands from external are too fast	The device cannot register the commands at this speed	Send commands with more time interval
2	Wait, the door-space is not free	The external supplier has not reported the door space as free	Report space free so the door can operate
3	Wait (press cycle time too short)	The pressing cycle interval is too fast	The machine can perform one pressing cycle every 30 seconds
4	PGA is in the wrong mode	The pressing mode is wrong for the pressing command	Change the pressing command or the pressing mode
5	Wait until the door drives are rebooted	Door drives must reboot due to safety	Wait a few seconds until the door drives can execute commands again
6	PGA is not ready or not enabled	The device is not ready or set enabled	Check if the device is ready and enabled
7	PGA is not Init	The PGA is not Initialized after safety or reset	After safety or reset, the PGA is not initialized. Enable and clear, it will do it automatically.
8	VFD is in safety mode or has an error. More information is displayed on the VFD.	More information about the VFD can be read from the display on it.	Check the PGA, restart and try again.
9	Press >>clear<< to continue	Process is stopped due to safety	Confirm with clear to continue the process

8. Resources and links

8.1. powRgrip ®

https://ch.rego-fix.com/en/products/system/powrgrip

8.2. OPC UA

https://opcfoundation.org/
https://opcfoundation.github.io/UA-.NETStandard/

- Github:
https://github.com/OPCFoundation/UA-Java-Legacy/blob/master/examples/basic/README.md
- Codesys:
https://de.codesys.com/produkte/codesys-runtime/opc-ua.html
- Rockwell:
https://www.rockwellautomation.com/docs/en/factorytalk-optix/1-00/contents-ditamap/developing-solutions/object-examples/opc-ua-client-example.html
- Siemens:
https://support.industry.siemens.com/cs/document/109762770/s7-anwenderbaustein-f%C3%BCr-den-opc-ua-client-einer-simatic-s7-1500-?lc=de-de
- Schneider Electric:
https://product-help.schneider-electric.com/Machine%20Expert/V1.2/de/m262prg/m262prg/OPC_UA_Server_Client_Configuration/OPC_UA_Server_Client_Configuration-1.htm?rhtocid=_0_18_0
- Fanuc:
https://www.fanuc.eu/ch/de/cnc/connectivity/opc-server
- Omron:
https://industrial.omron.eu/en/solutions/product-solutions/nj5-controller-with-opc-ua

9. List of abbreviations

Abbreviation	Description
APG	Adapter for powRgrip ®. Insert for clamping/releasing tool holders.
API	Application Programming Interface
OPC UA	Open Platform Communications Unified Architecture. Open-source communication protocol standard for data exchange.
PGA	Automated clamping unit for powRgrip ® toolholders
PG	powRgrip ®
UaExpert ®	OPC UA client
VFD	Variable frequency drive

10. Attachments